UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/804,346 | 03/12/2001 | Kenneth A. Chupa | CA920010012US1 | 1135 |

7590 09/23/2004

| EXAMINER |
|---|
| YIGDALL, MICHAEL J |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2122 | |

A. Bruce Clay
IBM Corporation
T81/062
PO Box 12195
Research Triangle Park, NC 27709

DATE MAILED: 09/23/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/804,346 | CHUPA ET AL. | |
| | Examiner | Art Unit | |
| | Michael J. Yigdall | 2122 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *25 June 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. This Office action is in reply to Applicant's response and amendment filed June 25, 2004. Claims 1-22 remain pending.

### *Response to Arguments*

2. Applicant's arguments have been fully considered but they are not persuasive.

Applicant submits that Goldberg does not disclose a generator dictionary comprising at least one logical generator and at least one physical code generator, wherein the at least one logical generator calls the at least one physical generator to generate source code (page 10, second paragraph). Specifically, Applicant contends that Goldberg fails to disclose a code generator that can call another code generator to generate source code, in that the class hierarchy of Goldberg is not based on whether a code generator (logical generator) can call another code generator (logical generator or physical code generator) to generate source code.

However, the examiner respectfully disagrees with Applicant's characterization of the reference. The class hierarchy disclosed by Goldberg comprises code generators that derive or "call" other code generators (see FIG. 8). For example, the QueryObjectImplGenerator base class, a logical generator, may call the JavaQueryObjectImplGenerator subclass, another logical generator, which in turn may call the JDBCJavaQueryObjectImplGenerator subclass (see column 12, line 62 to column 13, line 20). The classes at the lowest level of the hierarchy, such as the JDBCJavaQueryObjectImplGenerator class and the ones listed in the exemplary generator dictionary (see column 12, lines 32-50), represent the physical code generator objects that generate source code, as acknowledged by Applicant (with regard to column 12, lines 10-12).

Thus, the class hierarchy disclosed by Goldberg embodies a logical generator that calls a physical code generator to generate source code.

Applicant further contends that the combination of Goldberg and Haikin does not disclose or suggest modifying the generator dictionary to associate a second code generator with said generator routine (page 10, bottom to page 11, second paragraph).

However, Goldberg alone discloses this feature, as determined upon a further reading of the reference and as presented in the claim rejections below. Specifically, Goldberg discloses adding new code generator subclasses to the class hierarchy to accommodate other database systems (see column 13, lines 21-26). The new physical code generators are in this manner associated with the higher-level generators in the hierarchy. Furthermore, the generator dictionary (see column 12, lines 32-50) must be modified so that these new generator objects may be used.

***Claim Rejections - 35 USC § 102***

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. Claims 1-22 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Pat. No. 6,496,833 to Goldberg et al. (art of record; herein "Goldberg").

With respect to claim 1 (currently amended), Goldberg discloses a computer system for generating source code (see the title and abstract), said computer system comprising:

(a) a generator dictionary associating a generator routine with a generator identity, said generator identity identifying a code generator (see column 12, lines 32-50, which shows a programming construct or dictionary that associates a generator routine for a specific language and database with a generator class name or identity that identifies a code generator) and said generator dictionary comprising at least one logical generator and at least one physical code generator (see column 12, lines 18-27, which shows the QueryObjectImplGenerator base class, a logical generator, and implementation classes, i.e. physical code generators).

(b) a code generation framework tool wherein said code generation framework tool, responsive to a request for an invocation of said generator routine, invokes said code generator identified by said generator identity associated with said generator routine (see column 11, lines 21-26, which shows a generator tool responding to input, i.e. to a request, and invoking the identified code generator);

wherein the at least one logical generator calls the at least one physical code generator to generate source code (see FIG. 8 and column 12, line 62 to column 13, line 20, which shows a logical generator such as JavaQueryObjectImplGenerator deriving or calling a physical code generator such as JDBCJavaQueryObjectImplGenerator to generate source code).

With respect to claim 2 (original), Goldberg further discloses the limitation wherein said generator dictionary comprises a plurality of generator routines, each of said generator routines

associated with a generator identity (see FIG. 8 and column 12, lines 32-50, which shows a plurality of generator routines associated with a generator identity).

With respect to claim 3 (original), Goldberg further discloses the limitation wherein said generator dictionary comprises a text file (see column 12, lines 32-50, which shows that the dictionary is in the form of source code, inherently comprising a text file).

With respect to claim 4 (original), Goldberg further discloses the limitation wherein said generator routine comprises a logical generator name (see column 12, lines 32-50, which shows that the generator routine comprises a logical generator name such as "sybase_ctlib" or "oracle_oci").

With respect to claim 5 (original), Goldberg further discloses the limitation wherein said code generation framework tool retrieves from said generator dictionary said generator identity responsive to said request (see column 11, lines 21-26, which shows that the generator tool selects the appropriate generator in response to the request).

With respect to claim 6 (currently amended), Goldberg discloses a method for generating source code from input data (see the title and abstract), said method comprising:

(a) responsive to a request for invoking a generator routine, identifying a code generator associated with said generator routine (see column 11, lines 21-26, which shows selecting the appropriate generator in response to a user request);

(b) passing said input data to said code generator identified (see column 11, lines 7-20, which shows input data specifying the operating environment, and column 12, lines 18-27, which shows passing the data to a generator method), said code generator being operable to:

call another code generator to generate the source code (see column 12, line 62 to column 13, line 20, which shows a generator such as JavaQueryObjectImplGenerator deriving or calling another generator such as JDBCJavaQueryObjectImplGenerator to generate source code); and

generate the source code (see column 11, lines 21-26, which shows generating source code).

With respect to claim 7 (original), Goldberg further discloses the limitation wherein said identifying comprises retrieving from a generator dictionary code generator identity data associated with said generator routine (see column 12, lines 27-51, which shows a programming construct or dictionary for retrieving the identity of a code generator associated with a generator routine that represents a specific language and database).

With respect to claim 8 (original), Goldberg further discloses the limitation wherein said identifying further comprises prior to said retrieving, locating said generator routine in said generator dictionary (see column 12, lines 32-50, which shows a "switch" construct, which inherently involves locating the appropriate "case" statement before returning the identity of a code generator).

With respect to claim 9 (original), Goldberg further discloses the limitation wherein said generator dictionary comprises a lookup table (see column 12, lines 32-50, which shows a programming construct or dictionary that serves as a lookup table).

With respect to claim 10 (original), Goldberg further discloses the limitation wherein said generator dictionary comprises a text file (see column 12, lines 32-50, which shows that the dictionary is in the form of source code, inherently comprising a text file).

With respect to claim 11 (original), Goldberg discloses a method of generating source code for a first and a second deployment environment from a single input (see FIG. 8 and column 12, lines 9-51, which shows code generators for generating source code for a plurality of platforms or deployment environments), said method comprising:

(a) invoking a first code generator to generate source code for said first deployment environment from said single input, said first code generator identified by retrieving code generator identity data from a generator dictionary based on a generator routine (see column 11, lines 21-26, which shows invoking the appropriate code generator based on input information, and column 12, lines 18-50, which shows retrieving the identity of a code generator from a generator dictionary based on a generator routine for the deployment environment, i.e. the first deployment environment).

(b) modifying said generator dictionary to associate a second code generator with said generator routine (see column 13, lines 21-26, which shows adding new code generator classes, i.e. modifying the generator dictionary, to associate other code generators).

(c) invoking said second code generator to generate source code for said second deployment environment from said single input, said second code generator identified by retrieving code generator identity data from said generator dictionary based on said generator routine (see column 11, lines 21-26, which shows invoking the appropriate code generator based

on input information, and column 12, lines 18-50, which shows retrieving the identity of a code generator from the generator dictionary based on a generator routine for the deployment environment, i.e. the second deployment environment).

With respect to claim 12 (original), Goldberg further discloses the limitation wherein said invoking said first code generator comprises a call issued by one of a code generation framework tool and a code generator; and wherein said invoking said first code generator comprises a call issued by one of said code generation framework tool and a code generator (see column 11, lines 21-26, which shows invoking or calling the appropriate code generator).

With respect to claim 13 (original), Goldberg further discloses the limitation wherein said modifying comprises editing said generator dictionary (note that modifying the generator dictionary inherently comprises editing the generator dictionary).

With respect to claim 14 (currently amended), Goldberg discloses a generator dictionary stored on a recordable medium comprising:

at least one logical generator and at least one physical code generator (see column 12, lines 18-27, which shows the QueryObjectImplGenerator base class, a logical generator, and implementation classes, i.e. physical code generators) and a plurality of generator routines, each of said generator routines associated with code generator identity data (see FIG. 8 and column 12, lines 32-50, which shows a programming construct or dictionary comprising a plurality of generator routines for specific languages and databases associated with code generator class names or identities), and

wherein the at least one logical generator calls the at least one physical code generator to generate source code (see FIG. 8 and column 12, line 62 to column 13, line 20, which shows a logical generator such as JavaQueryObjectImplGenerator deriving or calling a physical code generator such as JDBCJavaQueryObjectImplGenerator to generate source code).

With respect to claim 15 (currently amended), Goldberg discloses a code generation framework tool (see the title and abstract) comprising:

(a) a receiver for receiving input data (see GUI 634 in FIG. 6 and column 10, lines 47-53, which shows an interface or receiver for receiving input data);

(b) a generator dictionary accessor for retrieving data from a generator dictionary comprising at least one logical generator and at least one physical code generator (see column 12, lines 29-51, which shows an accessor method for retrieving data from a generator dictionary, and column 12, lines 18-27, which shows the QueryObjectImplGenerator base class, a logical generator, and implementation classes, i.e. physical code generators); and

(c) an invoking mechanism for calling a code generator (see code generator 604 in FIG. 6 and column 11, lines 21-26, which shows invoking a code generator); and

wherein, responsive to a receipt of input data at said receiving, said invoking mechanism calls a code generator identified by identity data retrieved by said generator dictionary accessor from a generator dictionary (see column 11, lines 21-26, which shows invoking a code generator in response to input data, and column 12, lines 18-51, which shows that the identity of a code generator is retrieved from the accessor method).

With respect to claim 16 (original), Goldberg further discloses a data dictionary associating a generator routine with identity data identifying a code generator (see column 12, lines 9-51, which shows a programming construct or dictionary that associates a generator routine for a specific language and database with a generator class name or identity, which identifies a code generator).

With respect to claim 17 (currently amended), Goldberg further discloses the limitation wherein said generator dictionary accessor identifies a generator routine within said input data received and wherein said code generator identified is determined by retrieving said identity data associated with said generator routine identified (see column 12, lines 29-51, which shows the accessor method for identifying a code generator based on a generator routine, which is determined from the input data specifying a language and database).

With respect to claim 18 (currently amended), Goldberg discloses a computer readable medium storing instructions and data (see column 20, lines 5-11), said instructions and data for adapting a computer system to:

(a) responsive to a request for invoking a generator routine, identify, in a generator dictionary that includes at least one logical generator and at least one physical code generator, a code generator associated with said generator routine (see column 11, lines 21-26, which shows selecting the appropriate generator in response to a user request, and column 12, lines 18-50, which shows identifying the code generator from a generator dictionary, including the QueryObjectImplGenerator base class, a logical generator, and implementation classes, i.e. physical code generators);

(b) pass said input data to said code generator identified (see column 11, lines 7-20, which shows input data specifying the operating environment, and column 12, lines 18-27, which shows passing the data to a generator method), said code generator being operable to:

call another code generator to generate the source code (see column 12, line 62 to column 13, line 20, which shows a generator such as JavaQueryObjectImplGenerator deriving or calling another generator such as JDBCJavaQueryObjectImplGenerator to generate source code); and

generate the source code (see column 11, lines 21-26, which shows generating source code).

With respect to claim 19 (currently amended), the limitations recited in the claim are analogous to those of claim 7 (see the reasoning applied to claim 7 above).

With respect to claim 20 (currently amended), the limitations recited in the claim are analogous to those of claim 8 (see the reasoning applied to claim 8 above).

With respect to claim 21 (currently amended), the limitations recited in the claim are analogous to those of claim 9 (see the reasoning applied to claim 9 above).

With respect to claim 22 (currently amended), the limitations recited in the claim are analogous to those of claim 10 (see the reasoning applied to claim 10 above).

***Conclusion***

5. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (703) 305-0352. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

After October 25, 2004, the examiner can be reached at (571) 272-3707, and the examiner's supervisor, Tuan Q. Dam can be reached at (571) 272-3694.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2122

mjy

ANTONY NGUYEN-BA
PRIMARY EXAMINER